

On the Joint Source-Channel Decoding of Variable-Length Encoded Sources: The Additive-Markov Case

K. P. Subbalakshmi, *Member, IEEE*, and Jacques Vaisey, *Member, IEEE*

Abstract—We propose an optimal joint source-channel maximum *a posteriori* probability decoder for variable-length encoded sources transmitted over a wireless channel, modeled as an additive-Markov channel. The state space introduced by the authors in a previous paper is used to take care of the unique challenges posed by variable-length codes. Simulations demonstrate that this decoder performs substantially better than the standard Huffman decoder for a simple test source and is robust to inaccuracies in channel statistics estimates. The proposed algorithm also compares favorably to a standard forward error correction-based system.

Index Terms—Bursty channels, error-resilient communication, joint source-channel decoding (JSCC), maximum *a posteriori* (MAP) decoding, variable-length codes.

I. INTRODUCTION

THE observation that Shannon's source-channel separation [2] theorem holds only under asymptotic conditions (arbitrarily large codeword lengths and large delays) has led to increased interest in exploring joint paradigms for source and channel coding. These joint source-channel coding (JSCC) schemes can be broadly classified into three different categories: joint source-channel encoding (JSCE) [3], [4]; joint source-channel decoding (JSCD) [5]–[9]; and rate allocation strategies [10], [11]. As the names suggest, these deal with the joint design of encoders and decoders and rate allocation between the channel and source codes, respectively.

Our current focus is to develop a JSCD scheme for variable-length encoded sources transmitted over a channel with memory. Previous work on JSCCs for channels with memory includes [4] and [12]. Of these, the first paper describes a JCSE technique and the second proposes a JSCD for a binary source. Our work differs from [12] in that we deal with variable-length encoded sources. To the best of our knowledge, this work is the first attempt to design an optimal decoder for *variable-length* encoded sources operating over *bursty channels*. Preliminary results of this study were published by us in [9].

Our goal is to achieve the best performance possible without the use of channel codes, which require overhead bits. This work is motivated by the facts that most real-world channels are not

memoryless and that (variable-length) entropy coding is generally required for a source-coder to achieve good rate-distortion performance. Although interleaving can be used to effectively convert a bursty channel into a binary symmetric channel (BSC), this strategy also results in delays, which may be unacceptable in some applications.

In the following, we describe the channel model and develop a maximum *a posteriori* (MAP) decoder for entropy-coded Markov sources transmitted over additive-Markov channels (AMC). The performance of this decoder is then evaluated by simulations using sample test sources, channels, and forward error-correcting codes (FEC). We also present the performance under mismatch conditions, demonstrating the decoder's robustness to errors in the estimation of the channel statistics.

II. THE MAP-AMC PROBLEM

This letter deals with the design of an optimal JSCD for variable-length encoded sources transmitted over bursty channels. Alajaji *et al.* developed a model for such a channel [13] in which the channel is described by

$$Y_i = X_i \oplus Z_i, \quad \forall i = 1, 2, \dots \quad (1)$$

where \oplus denotes the binary XOR operation and X_i , Z_i , and Y_i represent the binary random variables associated with bit position i of the input, the noise, and output random processes, respectively. The noise random process, Z , is assumed to be stationary, Markov, and independent of the input process. For a channel with error probability ϵ and correlation coefficient ρ_c , the transition probabilities $Q(z_n|z_{n-1}) \equiv \Pr\{Z_n = z_n | Z_{n-1} = z_{n-1}\}$ and the marginal probabilities of the noise bits $Q(z_n) \equiv \Pr\{Z_n = z_n\}$ are given by [13]

$$\begin{bmatrix} Q(0|0) & Q(1|0) \\ Q(0|1) & Q(1|1) \end{bmatrix} = \frac{1}{1+\delta} \begin{bmatrix} 1-\epsilon+\delta & \epsilon \\ 1-\epsilon & \epsilon+\delta \end{bmatrix}$$

with $Q(1) = \epsilon = 1 - Q(0)$ and where $\delta = \rho_c/(1 - \rho_c)$ is a correlation parameter.

Let us consider a stationary, variable-length encoded Markov source, transmitted as binary bits over an AMC. Let \mathbf{C} denote the set of all possible B -bit sequences of variable-length codewords. The j th such sequence is then denoted by $\mathbf{c}_j^{(j)} = \{c_{j,i}\}_{i=1}^{n(j)}$, where $c_{j,i}$ is the codeword corresponding to the i th symbol in the transmitted stream and $n(j)$ is the total number of codewords in this sequence. Similarly, $r_{j,i}$ is the i th symbol of the received bit stream under the same partition as $\mathbf{c}_j^{(j)}$. The problem is to find the most probable transmitted sequence given the received sequence, the source statistics, and the channel statistics. We consider the case of the first-order Markov source only (the extension to high-order

Paper approved by K. Rose, the Editor for Source-Channel Coding of the IEEE Communications Society. Manuscript received March 20, 2000; revised July 26, 2002 and February 18, 2003. This work was supported in part by the Natural Science and Engineering Research Council of Canada, in part by the Canadian TeleLearning-NCE, and in part by the New Jersey Center for Wireless Telecommunications. This paper was presented in part at the 33rd Annual Conference on Information Sciences and Systems, March 1999.

K. P. Subbalakshmi is with the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ 07030 USA (e-mail: ksubbala@stevens-tech.edu).

J. Vaisey is with the School of Engineering Science, Simon Fraser University, Burnaby, BC V5A 1S6, Canada (e-mail: vaisey@sfu.ca).

Digital Object Identifier 10.1109/TCOMM.2003.816944

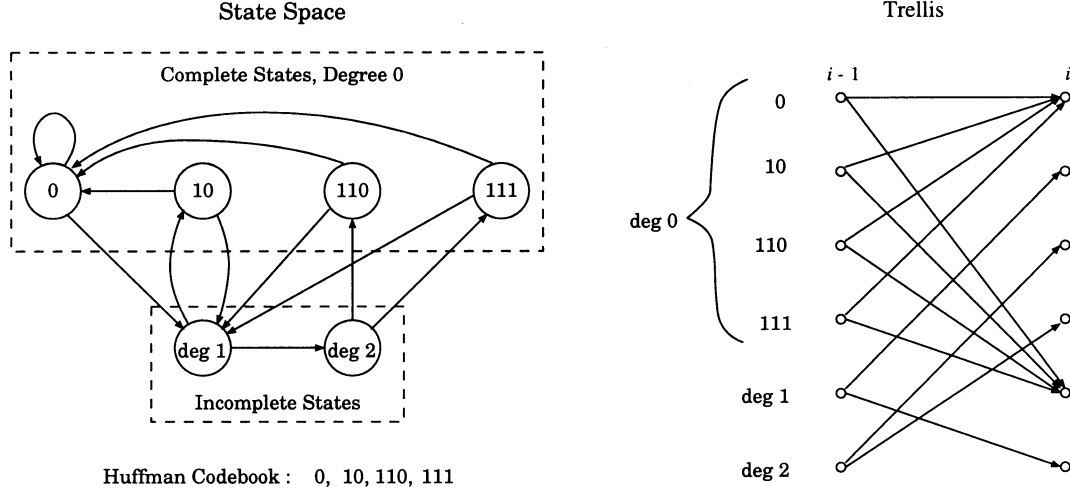


Fig. 1. MAP-AMC: Example state space and trellis.

sources is straightforward) and define \hat{j} to be the index of the most probable transmitted sequence. The MAP problem for the AMC (MAP-AMC) is then to determine the optimal sequence $\mathbf{c}_j^{n(\hat{j})} = \{c_{j,1}, \dots, c_{j,n(\hat{j})}\}$ according to

$$\mathbf{c}_j^{n(\hat{j})} = \arg \max_{\mathbf{c}_j^{n(j)}} \left\{ \Pr(c_{j,1}) \epsilon^{z_{j,1}} (1 - \epsilon)^{(1-z_{j,1})} \times \prod_{k=2}^{n(j)} [\Pr(c_{j,k} | c_{j,k-1})] \prod_{i=2}^B Q(z_{j,i} | z_{j,i-1}) \right\} \quad (2)$$

where $\Pr(c_{j,1})$ is the probability that codeword $c_{j,1}$ was transmitted first, $\Pr(c_{j,k} | c_{j,k-1})$ is the probability that the codeword $c_{j,k}$ was transmitted immediately after the codeword $c_{j,k-1}$, and $z_{j,i}$ is the i th noise bit under the j th partition, which is determined by the received bit stream, the specific partition, and (1). The maximization over j effectively searches through all possible error sequences and bit-stream partitions. This maximization can be time consuming and the problem is best solved by casting it into a dynamic programming framework; however, in order to accomplish this task, an efficient state space is needed. Before proceeding with the state-space development, we consider the right-hand side of (2), which can be factored into two terms: a channel term and a source term, with the former being

$$\epsilon^{z_{j,1}} (1 - \epsilon)^{(1-z_{j,1})} \prod_{i=2}^B Q(z_{j,i} | z_{j,i-1}).$$

In the dynamic program, every stage must know the noise bit of the previous stage in order to compute the continued product occurring in the channel term. This requirement translates into remembering the noise bit associated with the last bit of the codeword that immediately precedes the current node (state-stage pair) along the best metric path terminating at the current node. The solution is effectively formulated within the framework of the state space proposed by the authors for variable-length encoded Markov sources transmitted over the MAP-BSC [1], [14]. Note that since the state space is the same, the complexity of the MAP-AMC algorithm is essentially the same as

that of MAP-BSC and equal to the number of states. Only a small additional overhead is required for handling the noise bits.

The state space for the MAP-AMC decoder consists of two classes of states: the *complete* and the *incomplete* states. The decoder is said to be in a complete state if the most recently received bit is the last bit of a codeword. If this bit does not terminate a codeword, then the decoder is said to be in an incomplete state. The degree of incompleteness is simply the number of bits of an incomplete codeword that have been already received by the decoder. Fig. 1 shows the state space for a simple example, together with the possible transitions between stages $i-1$ and i . Each node in the trellis represents a state-stage pair. A special case of this state space was proposed in the context of variable-length dimension vector quantization in [15].

III. STATE SPACE AND THE PROPOSED ALGORITHM

The MAP-AMC decoding algorithm performs three main operations at each stage: the first examines the metrics of the paths entering each node in the trellis, the second looks for a path merger and declares codewords when merges are found, while the third uses the noise bits associated with predecessor nodes. For a complete node, this third task looks along the maximum-metric path and stores the last noise bit of the predecessor codeword; however, for incomplete nodes, a vector of values corresponding to the last noise bit of each of the complete nodes of the previous stage must be remembered.

In the ensuing discussion, we denote a node by an ordered pair of integers representing the state and stage indexes. For example, $\mathbf{v} = (v_x, v_y)$ represents the node corresponding to state v_x at stage v_y . The codeword dimensions are bounded between l_{\min} and l_{\max} . Thus, if there are N codewords in the codebook, then the complete states can be indexed from $1 \rightarrow N$ and the incomplete states from $(N+1) \rightarrow (N+l_{\max}-1)$. For each complete state, c_k , whose length we denote by d_k , and whose a th bit is denoted by $c_k(a)$, we compute the metric increment by looking back to the previous complete states, which are located d_k stages away. Each of these increments is denoted by $\mathcal{M}[k, i, c_j]$ and refers to the path segment that begins at node $(j, i-d_k)$ and terminates in the node (k, i) , where both k and j

are complete states. If $\mathbf{b}_{k,i}$ is the d_k most recently received bits at node (k,i) and $\mathbf{b}_{k,i}(a)$ denotes the a th bit in this segment, then we can write

$$\mathcal{M}[k,i,c_j] = \log_{10}(\Pr\{c_k|c_j\}) + \sum_{a=0}^{d_k-1} \log_{10} Q(z_{k,(i-a)}|z_{k,(i-a-1)})$$

where $z_{k,(i-a)} = \mathbf{b}_{k,i}(d_k-a) \oplus c_k(d_k-a)$ for $a \in \{0, \dots, d_k-1\}$ and $z_{k,i-d_k} = \mathcal{N}_{N+d_k-1}(j)$, which is 1 if the last bit of the codeword c_j was different from the corresponding received bit, and 0 otherwise. A formal expression for the meaning of \mathcal{N} will be presented below. Note that a is merely a running index and does not have any other significance.

Let $M_{k,i}$ be the metric value associated with the maximum-metric path terminating in (k,i) , where $k \in \{1, 2, \dots, N\}$ (i.e., the complete states). For incomplete states, we define $M_{k,i}(u)$ to be the u th element in the vector of metric values that need to be remembered at (k,i) , where $k \in \{N+1, N+2, \dots, N+l_{\max}-1\}$ and $u \in \{1, 2, \dots, N\}$. We note that updating $M_{k,i}(u)$ involves only a copy operation. Finally, let \mathbf{v}^p be the parent of node \mathbf{v} , where a parent is defined as the penultimate node in the maximum-metric path terminating at node \mathbf{v} , and let $d_{v_x^p}$ be the length of the codeword corresponding to the parent node (which is complete). The full algorithm can now be stated as follows.

Initialize:

Input first $l_{\min}-1$ bits
 For $i = 1, \dots, l_{\min}-1$,
 $M_{k,i} \leftarrow 0 \quad \forall k \in \{1, 2, \dots, N\}$.
 $M_{k,i}(u) \leftarrow 0 \quad \forall u \in \{1, 2, \dots, N\}; \quad \forall k \in \{N+1, \dots, N+l_{\max}-1\}$.
 $i \leftarrow l_{\min}$
 For $k = 1, 2, \dots, N$
 $\mathbf{v} = (k, i)$
 $\mathbf{v}^p = \Phi$, the empty set (no parents)

Input:

Input bit at the i th stage

Update path metrics:

For $k = 1, 2, \dots, N$
 see the equation at the bottom of the page

Update noise bits:

For $k = 1, 2, \dots, N$

$$\mathcal{N}_{k,i} \leftarrow \begin{cases} \mathbf{b}_{k,i}(d_k) \oplus c_k(d_k), & i \geq d_k \\ \text{Don't care}, & i < d_k \end{cases}$$

$$\mathcal{N}_{k,i}(u) \leftarrow \begin{cases} \mathcal{N}_{u,(i-1)} & \forall u \in \{1, 2, \dots, N\}, \\ & k = N+1. \\ \mathcal{N}_{(k-1),(i-1)}(u) & \forall u \in \{1, 2, \dots, N\}, \\ & \forall k \in \{N+2, \dots, N+l_{\max}-1\} \end{cases}$$

Update paths:

For $k \in \{1, 2, \dots, N\}$

$\mathbf{v} = (k, i)$

$v_x^p = j_k^*; \quad v_y^p = i - d_{j_k^*};$

Merge Check and Output:

For degree-zero nodes, we trace back from the respective nodes at stage i ; however, for any incomplete state k , we trace back from *all* of the complete states of stage $i - k + N$; this must be done because no parent node can be discarded at incomplete states, since it may be the parent of some state at a later stage.

Loop Back/Stop:

If $i < B$, Set $i \leftarrow i+1$ and go to **Input**
 Else Stop.

IV. EXPERIMENTAL RESULTS

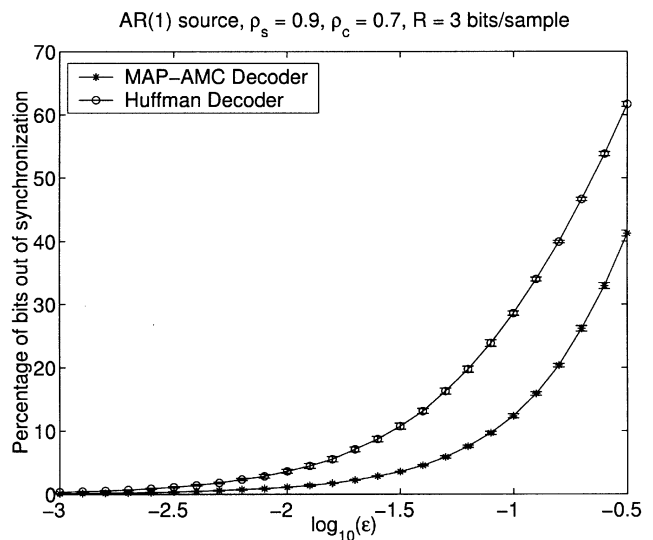
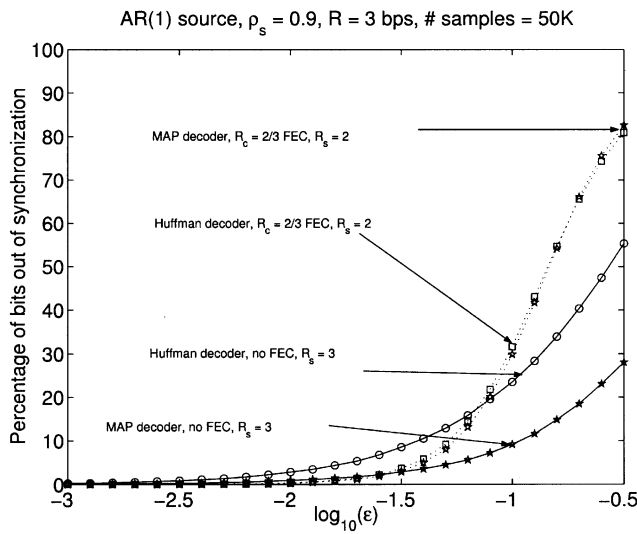
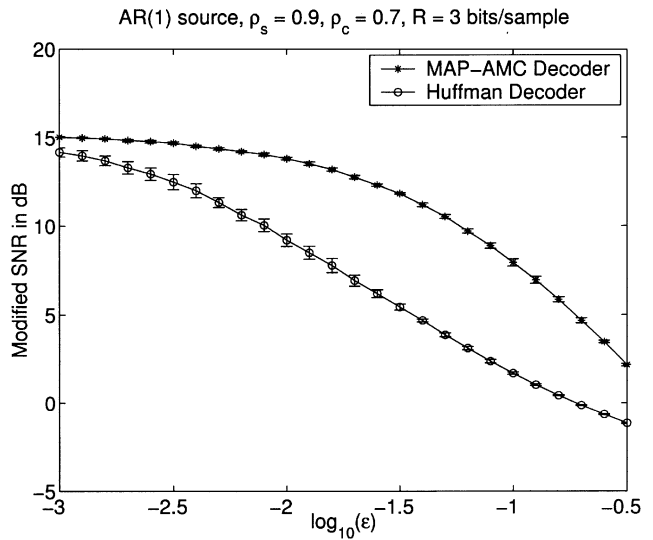
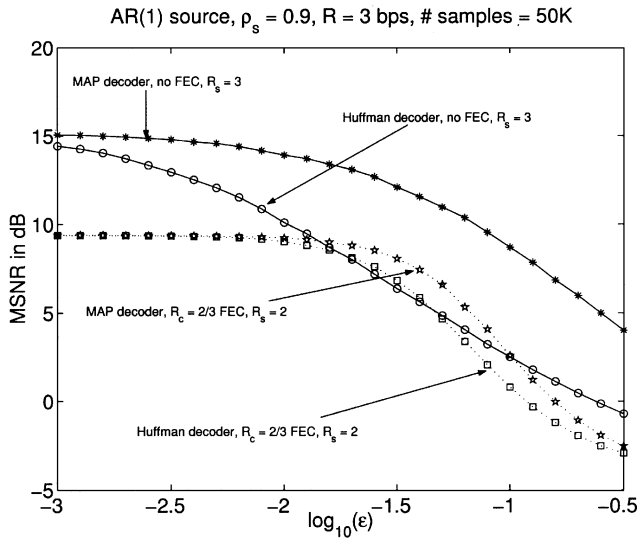
We define two measures of performance for the algorithms: the percentage of bits that are out of synchronization (PBOS) and the modified signal-to-noise ratio (MSNR). The PBOS captures the synchronization-loss aspect of the performance and is defined as the ratio of the number of bits that are received out of synchronization to the total number of bits in the stream. The MSNR is simply the SNR between the original, unquantized source and the synchronous portions of the decoded stream. These performance measures serve to somewhat decouple the effects of loss of synchronization from that of word errors and are appropriate for signals where the synchronization loss is less annoying. In general, performance measures could be application dependent. Although we present results for Huffman codes, our decoder is applicable to any entropy code that can be implemented as a table lookup algorithm.

Experiments were performed on 50 000 samples of zero-mean, first-order, Gauss-Markov sources (driven by a unit variance, Gaussian random process) with $\rho_s = 0.9$ and $\rho_s = 0.8$. Both sources were quantized using nine levels, with the

$$M_{k,i} \leftarrow \begin{cases} \max_j \{M_{(N+d_k-1),(i-1)}(j) + \mathcal{M}[k,i,c_j]\}, & i > d_k \\ \log_{10}(\Pr\{c_k\}) + z_{k,1} \log_{10}(\epsilon) + (1 - z_{k,1}) \log_{10}(1 - \epsilon) \\ \quad + \sum_{a=0}^{d_k-2} \log_{10} Q(z_{k,i-a}|z_{k,i-a-1}), & i = d_k \\ 0, & i < d_k \end{cases}$$

$$M_{k,i}(u) \leftarrow \begin{cases} M_{u,(i-1)} & \forall u \in \{1, 2, \dots, N\}, \quad k = N+1. \\ M_{(k-1),(i-1)}(u) & \forall u \in \{1, 2, \dots, N\}, \quad \forall k \in \{N+2, \dots, N+l_{\max}-1\} \end{cases}$$

$$j_k^* \leftarrow \arg \max_j \{M_{(N+d_k-1),(i-1)}(j) + \mathcal{M}[k,i,c_j]\}, \quad \forall k \in \{1, 2, \dots, N\}$$

Fig. 2. Performance comparisons: $\rho_s = 0.9, \rho_c = 0.8$.Fig. 3. Performance comparisons: $\rho_s = 0.9, \rho_c = 0.7$.

step sizes set to 1.25 and 0.9, respectively; the quantizers were chosen simply to give “reasonable” fidelity under error-free conditions. Once quantized, the sources were Huffman encoded and corrupted by random bit-error patterns representing different AMCs. The quantized data is modeled as a Markov(1) source and the MAP-AMC decoder developed above is applied to it. The statistics of the source were obtained from the quantized data using a training sequence. A counter was set up to count the number of occurrences of each event and the probability was calculated from that. The transition probability was also calculated by the “counting” method, where the total number of favorable transitions are counted. For the experiments with conventional FECs, we used a rate 2/3 convolutional code (with depth 100, block interleaving) having constraint length two, as in [16]. To keep the overall bit rate the same as in the basic case, we code the source at 2 bits/sample by using a five-level quantizer and Huffman codes. The channel bit rate is now altered due to the introduction of the FEC in the loop. Hence, in the experiment where the MAP decoder follows the FEC, we use the new channel bit-error rate in the MAP

decoder to ensure fair comparison. The results presented are an average of six different channel realizations (random seeds) and are depicted in Fig. 2 for $\rho_c = 0.8$ and $\rho_s = 0.9$.

From the plots, we see that the MAP-AMC performs better than the Huffman decoder at all error rates considered, with maximum improvement being 6.3 dB (MSNR) and 27.3% (PBOS). We can also see that the MAP-AMC does better than a scheme that involves a convolutional coder and interleaver. Experiments where the source correlation is reduced to $\rho_s = 0.8$ also show a similar pattern of improvement over the Huffman decoder, but with a reduced maximum improvement. This result is expected, since the redundancy due to the source memory has been reduced. We then looked at the effect of reducing the channel memory by lowering ρ_c to 0.7. Fig. 3 shows that the performance trend is the same as with $\rho_c = 0.8$, although the peak MSNR and PBOS improvements are now 6.67 dB and 20.87%, respectively. We then set the error rate to $\epsilon = 10^{-1.5}$ and swept ρ_c from 0.1 \rightarrow 0.9. The performance plots are shown in Fig. 4 and, interestingly, it is seen that the

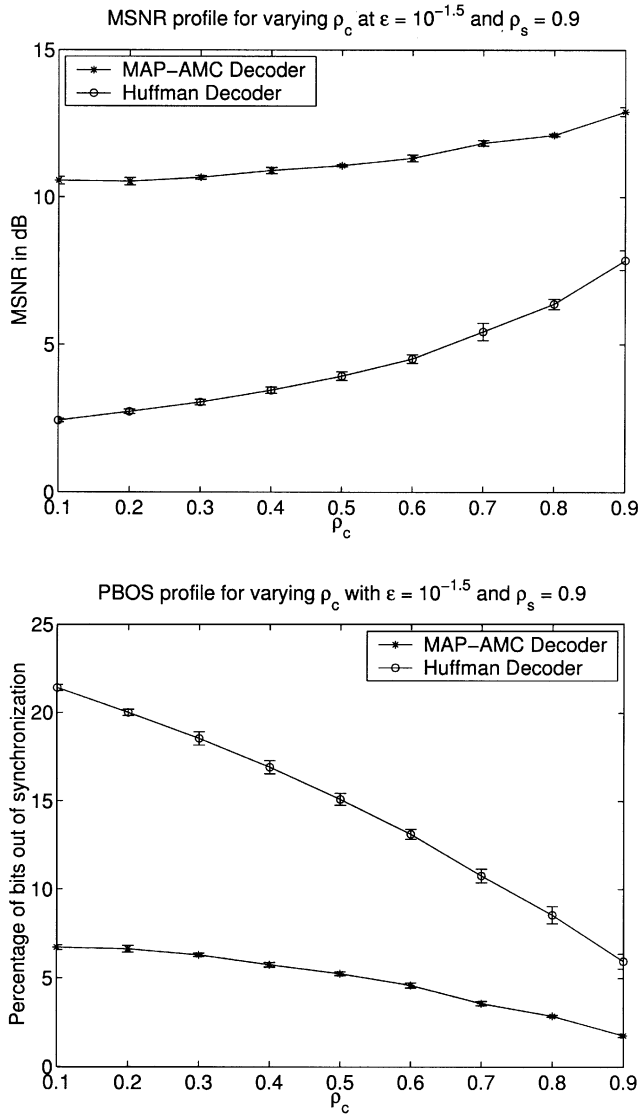


Fig. 4. Effect of channel memory: $\rho_s = 0.9$, $\epsilon = 10^{-1.5}$.

performance of *both* the MAP-AMC and Huffman decoders increases with the channel memory. Although somewhat surprising, the Huffman results can be explained by the fact that increasing the channel memory results in greater error clustering: the result is fewer symbols being corrupted and a smaller number of synchronization losses. We also see that the relative performance of the MAP-AMC decoder is always better than the Huffman decoder and actually increases as ρ_c is dropped. In general, it is difficult to separate the improvement of the MAP-AMC decoder over Huffman decoder into that due to source and channel memory. For example, we observed that setting $\rho_s = 0$ resulted in a MAP-AMC decoder that performs essentially the same as the Huffman decoder. Future work needs to be done to explain this phenomenon and to verify that it occurs under all operating conditions. As a final note, the $\rho_s = 0$ results show that the “believe what you see” rule (the decoder parses the received stream as is) is sometimes a good solution. In fact, Alajaji *et al.* [12] have derived conditions for the optimality of the “believe what you see” rule for the simple cases of binary-symmetric Markov and binary independent and

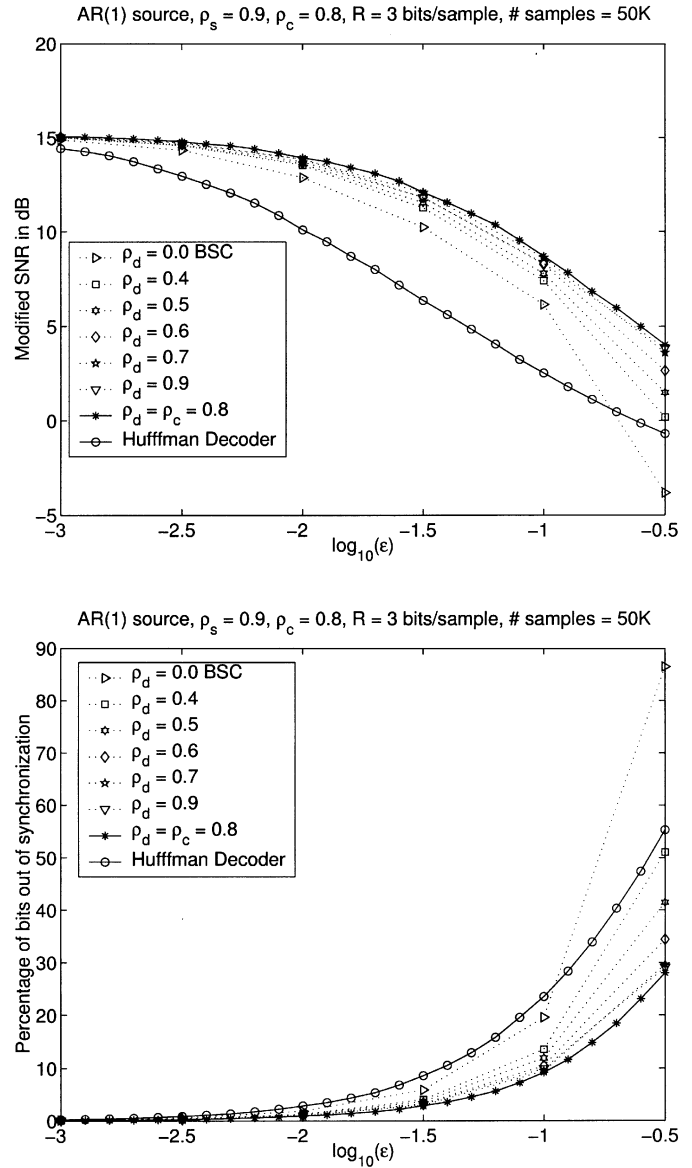


Fig. 5. Performance under channel mismatch: $\rho_s = 0.9$, $\rho_c = 0.8$.

identically distributed sources over an AMC. Derivations for more complicated sources, such as the one considered in this letter, are challenging problems and are not attempted here.

In practice, the channel estimates may be inaccurate or poor. Studies on the performance of our decoder to these mismatches show that the proposed decoder is very robust to channel mismatches. For example, we transmitted 50 000 samples of a $\rho_s = 0.9$ Gauss–Markov source through a channel with $\rho_c = 0.8$. The received data was then decoded using decoders designed for an “erroneous” channel correlation value of ρ_d ranging from $0 \rightarrow 0.9$, with $\rho_d = 0$ corresponding to an assumed BSC channel. The average results from the six different channel realizations are shown in Fig. 5, and it is seen that the MAP-AMC is reasonably robust to errors in estimating ρ_c and only a huge mismatch of 0.9 causes the MAP-AMC result to drop below the Huffman curve.

Finally, a note on computational complexity: each data point shown in Figs. 2–4 used approximately one minute of CPU time

on a SUN ULTRA 10 computer for the MAP-AMC decoder and about four seconds for the Huffman decoder.

V. CONCLUSIONS

This letter has examined the design of a joint source-channel MAP decoder for entropy-coded Markov sources transmitted over an AMC. The state space developed for the MAP-BSC for Markov sources [1] can be used in the dynamic programming formulation for the MAP-AMC. Simulations have demonstrated that this decoder does significantly better than the conventional one at all channel error rates considered, with these conclusions holding both with and without FEC. The decoder is robust to inaccuracies in the channel-correlation estimate. Finally, the overall performance of the MAP-AMC is seen to be significantly better than that of the MAP-BSC [1], indicating that it is critical to include the channel memory in the metric calculation. This improvement is not surprising, since channels with memory have higher capacity than those without [17].

ACKNOWLEDGMENT

The authors wish to thank Q. Chen for some of the simulation results that appear in this letter.

REFERENCES

- [1] K. Subbalakshmi and J. Vaisey, "On the joint source-channel decoding of variable-length encoded sources: The BSC case," *IEEE Trans. Commun.*, vol. 49, pp. 2052–2055, Dec. 2001.
- [2] S. Vembu, S. Verdu, and Y. Steinberg, "The source-channel separation theorem revisited," *IEEE Trans. Inform. Theory*, vol. 41, pp. 142–163, Jan. 1995.
- [3] J. G. Dunham and R. M. Gray, "Joint source and noisy channel trellis encoding," *IEEE Trans. Inform. Theory*, vol. IT-27, pp. 516–519, July 1981.
- [4] N. Phamdo, F. Alajaji, and N. Farvardin, "Quantization of memoryless and Gauss–Markov sources over binary Markov channels," *IEEE Trans. Commun.*, vol. 45, pp. 668–674, Jan. 1997.
- [5] N. Demir and K. Sayood, "Joint source/channel coding for variable length codes," in *Proc. IEEE Data Compression Conf.*, 1998, pp. 139–148.
- [6] R. Bauer and J. Hagenauer, "Iterative joint source/channel-decoding using reversible variable length codes," in *Proc. IEEE Data Compression Conf.*, Snowbird, UT, 2000, pp. 93–102.
- [7] A. Kopansky and M. Bystrom, "Sequential decoding of MPEG-4 coded bitstreams for error resilience," in *Proc. 33rd Annu. Conf. Information Sciences and Systems*, Mar. 1999, pp. 630–635.
- [8] M. Park and D. J. Miller, "Joint source-channel decoding for variable-length encoded data by exact and approximate MAP sequence estimation," *IEEE Trans. Commun.*, vol. 48, pp. 1–6, Jan. 2000.
- [9] K. P. Subbalakshmi and J. Vaisey, "Optimal decoding of entropy-coded Markov sources over channels with memory," in *Proc. 33rd Annual Conf. Information Sciences and Systems*, Mar. 1999, pp. 624–629.
- [10] M. Bystrom and J. Modestino, "Combined source-channel coding for transmission of H.263 coded video with trellis-coded modulation over a slow fading Rician channel," in *Proc. IEEE Int. Symp. Information Theory*, Aug. 1998, p. 12.
- [11] B. Hochwald and K. Zeger, "Tradeoff between source and channel coding," *IEEE Trans. Inform. Theory*, vol. 43, pp. 1412–1424, Sept. 1997.
- [12] F. Alajaji, N. Phamdo, N. Farvardin, and T. Fuja, "Detection of binary Markov sources over channels with additive Markov noise," *IEEE Trans. Inform. Theory*, vol. 42, pp. 230–239, Jan. 1996.
- [13] F. Alajaji and T. Fuja, "A communication channel modeled on contagion," *IEEE Trans. Inform. Theory*, vol. 40, pp. 2035–2041, Nov. 1994.
- [14] K. P. Subbalakshmi and J. Vaisey, "Joint source-channel decoding of entropy-coded Markov sources over binary symmetric channels," in *Proc. IEEE Int. Conf. Communications*, vol. 1, June 1999, pp. 446–450.
- [15] A. Makur and K. P. Subbalakshmi, "Variable dimension VQ encoding and codebook design," *IEEE Trans. Commun.*, vol. 45, pp. 897–900, Aug. 1997.
- [16] J. G. Proakis, *Digital Communications*, 4 ed. New York: McGraw-Hill, 2001.
- [17] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.